AD-A067 147 WHARTON SCHOOL PHILADELPHIA PA DEPT OF DECISION SCIENCES F/G 9/2
THE MULTIPLE PATH PROBLEM IN DATABASE SCHEMATA.(U)
JAN 79 O P BUNEMAN N00014-75-C-0462
UNCLASSIFIED 79-03-06 NL

| OF |
AD
A067147

END
DATE
FILMED
6-79
DDC

Wharton

Department of Decision Sciences

12

LEVEL II

University of
Pennsylvania
Philadelphia PA 19104

(12)

(6) THE MULTIPLE PATH PROBLEM IN DATABASE SCHEMATA,

(10) O. Peter/Buneman

(14) 79-03-06

(9) Technical rept. Apr 78-Mar 79,

(11) Jan 79

(12) 20p.

Department of Decision Sciences
The Wharton School
University of Pennsylvania
Philadelphia, PA 19104

Research supported in part by the Office of Naval Research under Contract N00014-75-C-0462

(15)

408 757

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: 79-03-06	2. GOVT ACCESSION NO.
3. RECIPIENT'S CATALOG NUMBER	
4. TITLE (and Subtitle): The Multiple Path Problem in Database Schemata	5. TYPE OF REPORT & PERIOD COVERED: Technical Report April 1978 - March 1979
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s): O. Peter Buneman	8. CONTRACT OR GRANT NUMBER(s): N00014-75-C-0462
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Department of Decision Sciences, University of Pennsylvania, Philadelphia, PA 19104	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: Task NR049-272
11. CONTROLLING OFFICE NAME AND ADDRESS: Office of Naval Research	12. REPORT DATE: January 1979
	13. NUMBER OF PAGES: 16
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)	15. SECURITY CLASS. (of this report): Unclassified
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

Distribution unlimited

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

databases, database schemata, query languages, functional dependencies

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

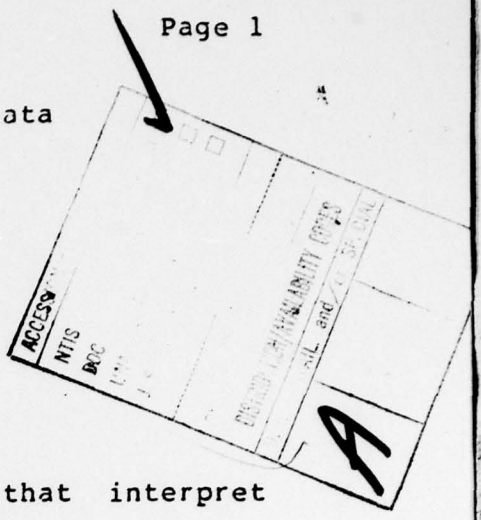
Certain database query systems, especially those that interpret natural language, do not require the user to have comprehensive knowledge of the database schema in order to form a meaningful query. Instead, the query is used to reference entities in the database schema and a path is automatically found through the schema which connects those entities. In this paper, we will classify those schemata in which all paths are equivalent, and then derive a method of marking a schema in order to determine the "natural" paths between two entities when different paths give rise to different results.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601 |

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

The Multiple Path Problem in Database Schemata

Peter Buneman



Introduction

Certain database query systems, especially those that interpret natural language [1, 2], do not require the user to have comprehensive knowledge of the database schema in order to form a meaningful query. Instead, the query is used to reference entities in the database schema and a path is automatically found through the schema which connects those entities. There is an obvious problem of choice when more than one such path exists. In this paper, we will classify those schemata in which all paths are equivalent, and then derive a method of marking a schema in order to determine the "natural" paths between two entities when different paths give rise to different results.

Functional Dependencies

In one sense or another, the notion of functional dependency is central to nearly every model of database semantics. A functional dependency describes the existence of a function between two classes of objects in the database which obtains even though those classes may vary in time. The importance of functional dependencies to the relational model has been fully treated by Codd and others [3, 4]. As another example, the set ownership construct of DBTG [5] structures

establishes, when the MANDATORY option is employed, a functional dependency of the owning class upon the owned class.

To start with, we shall assume a very simple model of a database schema: an acyclic graph of functional dependencies. For example figure 1 shows a set of dependencies which derive from

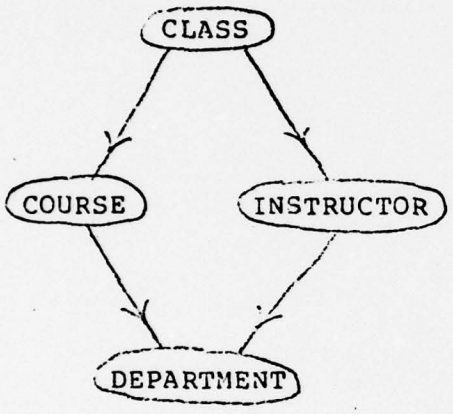


Figure 1. A set of functional dependencies

the observations that each CLASS is an instance of a COURSE and HAS an INSTRUCTOR. Each COURSE is offered in a DEPARTMENT and each INSTRUCTOR belongs to a department. Figure 2 shows a set of relational definitions which might be used to describe this set of dependencies

CLASS(HOUR, ROOM#, COURSE#, INSTRUCTOR#, ...)
INSTRUCTOR(INSTRUCTOR#, DEPT#, ...)
COURSE(COURSE#, DEPT#, ...)
DEPARTMENT(DEPT#,.....)

Figure 2.

Strictly speaking, the relational model does not contain any notion of the dependency of the tuples of one relation upon those of another, and only a partial dependency is given by the various natural joins. However an extension of the model by Smith and Smith [6] describes how functional dependencies may be used in the design of a relational database. For the Bachman diagram (DBTG schema) that corresponds to figure 1, one has simply to invert it.

. It is apparent that there are two paths from CLASS to DEPT in figure 1, and it may be that there is a constraint on the database that permits an instructor to teach only those courses which are offered in his department. That is, having specified a member of CLASS, the same member of DEPT is reached no matter which path is taken. We shall use the term natural to describe a diagram with this property. In order to demonstrate that natural diagrams do ,in fact, arise naturally, consider the schema in figure 3. In order to enroll in a course, a class must be specified. Moreover, the direct dependency of course upon enrollment is usually necessary to prevent a student enrolling in two sections of the same course.

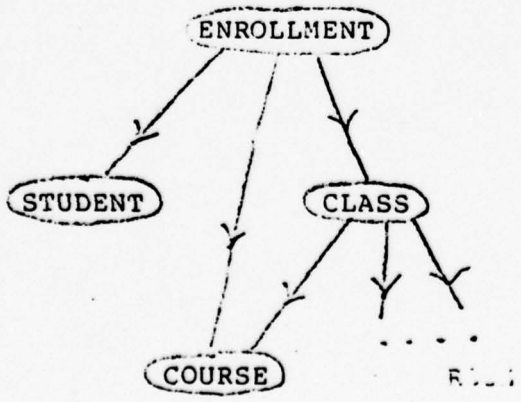


Figure 3.

Strictly speaking, the relational model does not contain any notion of the dependency of the tuples of one relation upon those of another, and only a partial dependency is given by the various natural joins. However an extension of the model by Smith and Smith [6] describes how functional dependencies may be used in the design of a relational database. For the Bachman diagram (DBTG schema) that corresponds to figure 1, one has simply to invert it.

. It is apparent that there are two paths from CLASS to DEPT in figure 1, and it may be that there is a constraint on the database that permits an instructor to teach only those courses which are offered in his department. That is, having specified a member of CLASS, the same member of DEPT is reached no matter which path is taken. We shall use the term natural to describe a diagram with this property. In order to demonstrate that natural diagrams do ,in fact, arise naturally, consider the schema in figure 3. In order to enroll in a course, a class must be specified. Moreover, the direct dependency of course upon enrollment is usually necessary to prevent a student enrolling in two sections of the same course.

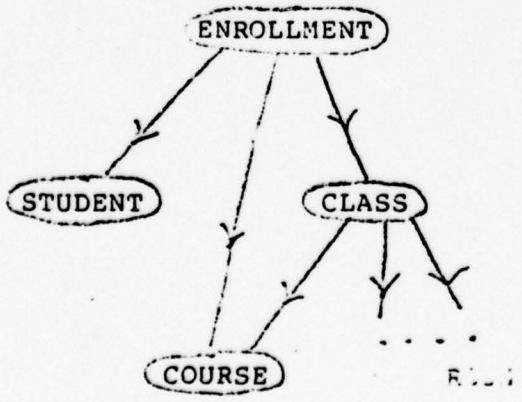


Figure 3.

Figure 3. A "natural" schema.

An extended notion of functional dependency has been studied by Armstrong [7] who gives axioms for what is termed a "full" set of functional dependencies. The relationship between functional dependencies satisfying these axioms and a normalized set of relations has been described by Beeri and by Risannen [8,9]. However, in the terms described above, these axioms require that the whole schema be natural and, with Date [10], we would argue that this requirement is for practical purposes too restrictive. Instead, we shall examine natural subschemata in a schema which is not itself natural. The reason for doing this is both as an aid in describing "natural" paths in the schema and also to investigate the problem of enforcing the integrity constraints imposed by natural subschemata. The formal results which follow are all, with minor modifications, applicable to schemata of partial functional dependencies. The only problem with discussing partial dependencies is that, even in a natural schema, not all downward paths are equivalent for, starting at a given object, some paths may not lead to a defined object.

Natural Schemata

We shall define a schema to be a collection of classes V, and functions F such that any f in F is a function $v_i \rightarrow v_j$ for v_i, v_j in V. With each schema there is an underlying directed graph whose arcs are the pairs (v_i, v_j) such that $f: v_i \rightarrow v_j$ is in F. We shall impose the additional restriction that the schema is acyclic, that is, its

graph contains no directed cycles. A subschema is defined as a subgraph: (V', F') is a subschema of (V, F) if $V' \subseteq V$ and $F' \subseteq F$. The terms "class" and "function" for a schema will be used interchageably with the terms "point" and "arc" for a directed graph.

Under what conditions is the union of two natural subschemata itself natural? We shall show that under certain conditions we need no further knowledge of the semantics of the database: this property may be inferred directly from graphical properties of the subschemata. Formally, a schema is natural if for any composition of functions $f_1 . f_2 \ldots f_m : u \to v$ and $g_1 . g_2 \ldots g_n : u \to v$,

$$f_1 . f_2 \ldots f_m(x) = g_1 . g_2 \ldots g_n(x)$$

for all $x \in u$. Let p be a point of a graph G and G' a subgraph of G. Define $dom(p,G',G)$ to be the set of points $u \in G'$ such that there is a directed path in G from u to p. Similarly let $sub(p,G',G)$ be the set of points v in G' for which there is a directed path from p to v in G. We shall say a set of points in G is connected if the induced subgraph on those points is connected.

Prop. 1. Let S_1 and S_2 be natural subschemata of S. If $dom(p, S_1 \cap S_2, S_i) \cap sub(p, S_1 \cap S_2, S_j)$ is connected in $S_1 \cap S_2$ for all points $p \in S_i$ and $q \in S_j$ $(i \neq j)$, then the union $S_1 \cup S_2$ is natural.

Proof. Without loss of generality, assume that $u \in S_1$, $v \in S_2$ and that there are two directed paths from u to v. These paths must both meet $S_1 \cap S_2$ in p_1 and p_2 which are in $H = dom(v, S_1 \cap S_2, S_2) \cap sub(u, S_1 \cap S_2, S_1)$,

and this subset is connected in $S_1 \cap S_2$ by the conditions of the prop. 2. Let $q_1, q_2 .. q_k$ be an (undirected) path in H which connects p_1 and p_2. Now make the inductive assumption that the subschema consisting of the path P_o from u to v via p_1 and any path P_i from u to v via q_i is natural. Let P_{i+1} be any path from u to v via q_{i+1} and assume, again w.l.o.g. that the direction of the arc between q_i and q_{i+1} is given by (q_i, q_{i+1}).

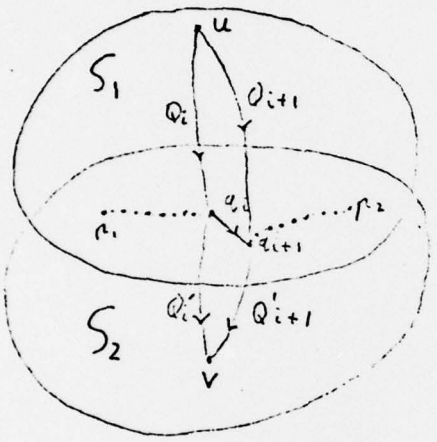


Figure 4

Letting Q_i denote the portion of P_i from u to q_i and Q'_i the portion from q to v and using the notation F(P) to denote the composition of functions associated with a directed path P, we have:

$$F(P_i) = F(Q_i).F(Q'_i) = F(Q_i).F((q_i, q_{i+1})).F(Q'_{i+1})$$

because S_1 is natural, and since S_2 is natural,

$$F(Q_i),F((q_i, q_{i+1})).F(Q'_{i+1}) = F(Q_{i+1}).F(Q'_{i+1}) = F(P'_{i+1})$$

Thus, by induction the paths from u to v via p_1 and p_2 (refer to figure 4) form a natural subschema, and the same is true for any pair of paths in $S_1 \cup S_2$. Although it is not hard to check that the conditions of prop. 1 are met, there are simple corollaries which may be used, in some circumstances, to make the check even simpler. In this context, we shall use the term tree to refer to a directed graph which is a tree (see Harary [11]) when the direction of the arcs is removed.

Cor. 1 If S_1 and S_2 are natural subschemata such that S and $S_1 \cap S_2$ is a tree, and such that $dom(p_i, S_1 \cap S_2, S_i)$ and $sub(p_i, S_1 \cap S_2, S_i)$ are trees for $p_i \in S_i$ (i=1,2), then $S_1 \cup S_2$ is natural. // The proof of this follows immediately from the fact that the intersection of two sub-trees in a tree is connected and thus the conditions of Prop 1. are met. This corollary in turn may be used to provide the simpler result:

Cor 2. If the intersection of two natural subschemata is a directed path, then the their union is also natural.

For example, in figure 5, if {1,2,3} is and {1,3,4} are natural subschemata, then so is {1,2,3,4} by cor. 2. If in addition we know that {2,3,4,5} is natural, cor. 1 ensures that the whole schema is natural.

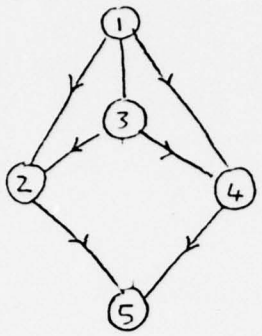


Figure 5

The ability to recognize natural subschemata is useful in querying the database, but their presence imposes additional constraints on the database which must be taken into account when performing updates. For example, consider the problem of adding a new member x to a class v in a natural schema S. It is necessary to specify the values of f(x) for any f: v -> u in S. In addition it is necessary to check that the addition of this member does not violate the naturalness of S. Fortunately, in many cases it is possible to limit the checking to a natural subschema of S. Let G be directed and acyclic and p be a root for G; that is there is a directed path from p to each point in G. We shall refer to a point q of G as essential if there is no connected subgraph whose removal disconnects p from q. BY the essential subschema for p, we mean the set of all points which are essential w.r.t. p or lie on a directed path from p to such a point.

Prop
~~Theorem~~ 2. Let p be the root of a subschema S, and let S' be its essential subschema. Then S and S' satisfy the conditions of prop. 1.

The proof of this result is straightforward, and its its usefulness is that we may limit consistency checks to the essential subschema of a class v when adding a new member to v. For example, when adding a member to class 1 in figure 5, it is only necessary to check that this update preserves naturalness of subschemata {1,2,3} and {1,3,4}. Modifications to f: u -> v should be checked for consistency in the essential subschemata for u, but note that such modifications are severely limited. For example, in if figure 1 is interpreted as a natural schema, modifications to the dependency CLASS -> INSTRUCTOR may only take place within the inverse image of some member of DEPARTMENT and that modifications to INSTRUCTOR -> DEPARTMENT are impossible when the inverse image of a member of INSTRUCTOR is non-empty. Natural schemata impose no restrictions on deletions, but as Smith and Smith point out [5], the removal of a member of u may require the removal of a substantial number of records in dom(u,S,S).

Describing Natural Paths

We now turn to the problem of describing natural paths through a schema which is not itself natural. If we now interpret figure 1, for example, as not being a natural schema, then the choice of which path to take between CLASS and DEPARTMENT is important: presumably the correct path is via COURSE. In this case, all paths except one are

natural. How are we to describe the natural path between two classes in a schema? One way is simply to keep a description of the appropriate path for each pair of classes in the schema. This solution, apart from being cumbersome to implement, places a considerable extra burden upon the database designer who must describe these paths. The solution proposed here is to mark those parts of the schema which are, in some sense, "unnatural". First we should note that we have not yet defined the term natural for a path. The natural paths from u to v cannot be intrinsically defined; they are simply the paths of choice in a query that given a member of u, requests a member of v. The only restriction that can be placed on natural paths is:

Condition 1. The union of all natural paths from u to v form a natural subschemata.

It is suggested that the characterization of natural paths may be performed as follows. Certain functions (arcs) in the schema are labelled with classes (points). If f: v -> w is labelled with u, this is interpreted as meaning that no natural paths from u may pass along the arc f. For example, in figure 1, the arc from INSTRUCTOR to DEPT may be marked with CLASS. This means that there is no natural path from CLASS to DEPT through INSTRUCTOR, however the elementary path from INSTRUCTOR to DEPT is itself natural. Note that only arcs below a given point in the schema are marked with u and that if the schema is marked with u wherever it is appropriate to do so, by condition 1 the union of all paths from u which do not traverse an arc labelled

with u, form a natural subschema.

Figure 6 shows a more complicated example of this kind of marking. A second condition arises from the desire to minimize the number of labels required to describe the unnatural paths.

Condition 2. If P is a path from v to w which is not natural, then any path from u to w with P as a final segment is also unnatural.

Thus, in figure 6, a path from ENROLLMENT via CLASS to DEPARTMENT which includes the arc from FACULTY to DEPARTMENT, cannot be considered natural because this arc is marked with CLASS.

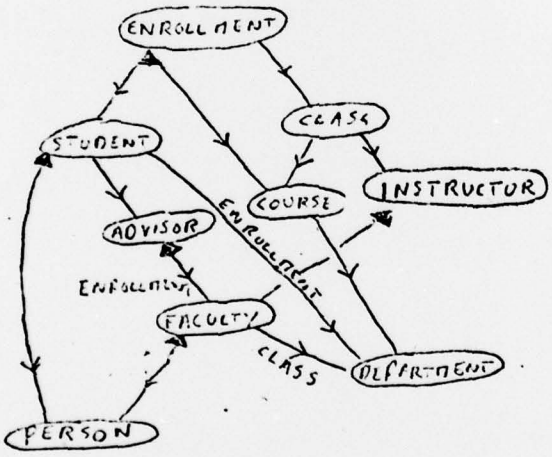


Figure 6

This marking can be readily used in any path finding algorithm which attempts to connect two points in the schema. If $u, u_1, u_2 \ldots u_n$ is being considered as an initial segment of a path from u to v and there is an arc from u_i to u_{i+1} then the path $u, u_1, u_2 \ldots u_n, u_{n+1}$ may

only be considered if the arc (u_n, u_{n+1}) is not labelled with u or u_i $(1<i<n)$. This characterization of a natural path means that there may be a natural subschema S which contains unnatural paths. For example the subschema on {STUDENT, ADVISER, FACULTY, DEPARTMENT} is natural, and trivially, prop. 1 allows us to add enrollment to this and preserve the naturalness of the subschema. But this subschema contains unnatural paths.

In concluding this section, it should be remarked that we have so far employed a particularly simple model of a database schema and that enrichment of this model is likely to add more techniques for the construction of natural paths. A particular case in point is that of keys. In terms of functional dependencies, suppose that u is a class in S and $f_1, f_2, \ldots f_m$ are (total) functions which respectively map u into $v_1, v_2, \ldots v_m$. We shall say that $\{f_1, f_2, \ldots f_m\}$ is a key for u if there exists a partial function

$$F: v_1 \times v_2 \times \ldots \times v_m \rightarrow u$$

such that

1. $F(f_1(x), f_2(x), \ldots f_m(x)) = x$ for all x in u.

2. There is no subset of $\{f_1, f_2, \ldots f_m\}$ with this property.

In figure 6, keys have been marked with a solid triangle. A key function, one which is a member of a key, defines a close relation between classes in the schema. In particular, key functions appear to be good candidates for inclusion in natural paths. WHile it is quite easy to construct examples in which there is a path through key

functions which is not natural, the following hypothesis is suggested.

Hypothesis. If there is a natural path from u to v in S and there is also a path from u to v which contains only key functions, then there is a natural path which contains only key functions.

Further Problems

In figure 6, the key function from STUDENT to PERSON, indicates that a student "is a kind of" person. However, as it stands, this diagram does not tell us whether a student can also be a faculty member, in other words we have no way of knowing whether the inverse images of STUDENT and FACULTY in PERSON have a non-empty intersection. Smith and Smith[12] have also described the concept of generalization in which data values may be used in cases where the intersection is empty, to indicate to which inverse image an object belongs. Such knowledge is clearly of great value in determining a natural path, but now introduces the complication that the natural path required by a query may depend on the data associated with that query and may not be found by inspection of the schema alone.

A second problem lies in the treatment of paths which go "up and down" the schema. In this paper only directed paths have been treated, however, database queries may involve traversing the schema in more than one direction. For example, a path between INSTRUCTOR and COURSE could be asked for from figure 1. In this case, the path of choice would require going up to CLASS and down to COURSE. While

the techniques presented here may greatly reduce the number of choices for such paths, even in a natural subschema, different undirected paths will produce different results. Some additional constructs are needed to resolve the choice in these circumstances.

Acknowledgement

The author is most grateful for several valuable discussions with Jon Hayward, Jerry Kaplan and Ron Lee.

References

1. Gerritsen, R., D.J. Root, J. Buchanan, Automated Database Programming, Working Paper 77-07-05, Wharton School, University of Pennsylvania (1977)

2. Sacerdoti, E.D. Language access to distributed data with error recovery, Technical Note TN140, Stanford Research Institute, (1977).

3. Codd, E.F. Further normalization of the relational model. In Courant Computer Science Symposium 6: Data Base Systems. Prentice Hall, (1971).

4. Codd, E.F. Recent investigations in relational database systems. Information Processing 74, 1017-1021, North-Holland (1974)

5. CODASYL Data Base Task Group Report. ACM, New York (1971)

6. Smith, J.M and D.C.P. Smith, Database Abstractions: aggregation, Comm. A.C.M., 20, 405-413 (June 1977)

7. Armstrong, W.W. Dependency structures for data base relationships. Information Processing 74, 580-583, North-Holland (1974)

8. Beeri, C., R. Fagin, and J.H. Howard. A complete axiomatization of functional and multivalued dependencies in database relations. ACM SIGMOD Int. Conf. on Management of Data, 47-61 Toronto (1977).

9. Risannen, J. Independent Components of Relations, ACM Transactions on Database Systems, 2, 4, 317-325 ((1978).

10. Date, C.J. An Introduction to Database Systems. 2nd edition, Addison-Wesley (1977).

11. Harary, F. Graph Theory Addison-Wesley, 1969.

12. Smith, J.M. and D.C.P. Smith, Database Abstractions: Aggregation and Generalization. ACM Transactions on Database Systems. 2, 105-133 (1977).

DISTRIBUTION LIST

Department of the Navy - Office of Naval Research

Data Base Management Systems Project

Defense Documentation
Cameron Station
Alexandria, VA 22314
12 copies

Office of Naval Research
Branch Office, Chicago
536 South Clark Street
Chicago, IL 60605

Office of Naval Research
New York Area Office
715 Broadway - 5th Floor
New York, N.Y. 10003

Dr. A.L. Slafkosky
Scientific Advisor (RD-1)
Commandant of the Marine Corps
Washington D.C. 20380

Office of Naval Research
Code 458
Arlington, VA 22217

Office of Naval Research
Information Systems Program
Code 437
Arlington, VA 22217
2 copies

Office of Naval Research
Branch Office, Boston
495 Summer Street
Boston, MA 02210

Office of Naval Research
Branch Office, Pasadena
1030 East Green Street
Pasadena, CA 91106

Naval Research Laboratory
Technical Information Division
Code 2627
Washington, D.C. 20375
6 copies

Office of Naval Research
Code 455
Arlington, VA 2217

Naval Electronics Lab. Cener
Advanced Software Technology Div.
Code 5200
San Diego, CA 92152

Mr. E. H. Gleissner
Naval Ship Research and
Development Center
Computation and Mathematic Dept.
Bethesda, MD 20084

Mr. Kim Thompson
Technical director
Information Systems Division
OP-911G
Office of Chief Naval Operations
Washington, D.C. 20350

Prof. Omar Wing
Columbia University
in the City of New York
Dept. of Electrical Engineering
and Computer Science
New York, N.Y. 10027

Commander, Naval Sea Systems Command
Department of the Navy
Washington, D.C. 20362
ATTENTION: PMS30611

Captain Richard Martin, USN
Commanding Officer
USS Francis Marion (LPA-249)
FPO New York 09501

Captain Grace M. Hopper
NAICOM/MIS Planning Branch
OP-916D
Office of Chief of Naval Research
Washington, D.C. 20350

Bureau of Library and
Information Science Research
Rutgers - The State University
189 College Avenue
New Brunswick, N.J. 08903
ATTENTION: Dr. Henry Voos

Defense Mapping Agency
Topographic Center
ATTN: Advanced Technology Div.
Code 41300
6500 Brookes Lane
Washington, D.C. 20315